Sprawozdanie nr 3
Praktyka i teoria szeregowania zadań

Data: 14.01.2021
Autor: Damian Tabaczyński
Indeks: inf136819
Grupa dziekańska: L6 (przeniesienie z L5)
Zajęcia: czwartek 11:45

1. Opis problemu

Problemem rozważanym w niniejszym sprawozdaniu jest problem szeregowania **n** zadań w systemie przepływowym z j. angielskiego “flow shop”, składającego się z trzech maszyn dedykowanych. Głównym celem jest znalezienie uszeregowania zadań przy minimalizacji średniego ważonego opóźnienia $\mathbf{D_w}$ wyrażonego wzorem**:**

$$D_W = \frac{\sum_{j=1}^{n} wj * Dj}{\sum_{j=1}^{n} wj}$$

Zagadnienie to można przedstawić również w notacji trójpolowej:

$$\mathbf{F3 \mid \mid D_W}$$

Ważną cechą tego problemu w porównaniu do poprzednich sprawozdań jest podział każdego zadania na trzy osobne operację, które muszą być wykonane przez trzy różne maszyny dedykowane. Wykonywanie zadań odbywa się bez możliwości ich przerwania. Niezbędnym założeniem jest by każda maszyna wykonywała tylko jedno zadanie i jedno zadanie było wykonywane przez jedną maszynę w tej samej jednostce czasu. Dodatkowo wygenerowane rozwiązania muszą być rozwiązaniami permutacyjnymi.

Zgodnie z nomenklaturą podaną w zadaniu oraz przedstawioną na wykładzie:

- każde zadanie $\mathbf{J_j}$ opisane jest trzema czasami trwania operacji danego zadania tzn. $\mathbf{p_{j1}}$, $\mathbf{p_{j2}}$, $\mathbf{p_{j3}}$ realizowanych odpowiednio na maszynach $\mathbf{M_1}$, $\mathbf{M_2}$, $\mathbf{M_3}$.
- każde zadanie $\mathbf{J_j}$ posiada oczekiwany termin zakończenia wykonywanie $\mathbf{d_j}$ oraz wagę $\mathbf{w_j}$
- $\mathbf{C_j}$ oznacza moment zakończenia wykonywania zadania $\mathbf{J_j}$ w uszeregowaniu

- $\mathbf{D_j}$ oznacza oznacza opóźnienie dla zadania $\mathbf{J_j}$ i wynosi $\mathbf{D_j} = \max\{0, \mathbf{C_j} - \mathbf{d_j}\}$

W ramach rozwiązania należy opracować i przedstawić generator instancji oraz algorytm listowy rozwiązujący przedstawiony problem, który nie przekracza złożoności obliczeniowej $\mathbf{O(n^2 \log n)}$.

2. Opis generatora instancji

W całym algorytmie generowania instancji zmienna:

- **max_length** oznacza maksymalną wartość sumy wszystkich operacji zadania i wynosi **100**
- **max_weight** oznacza maksymalną wartość $\mathbf{w_j}$ zadania i wynosi **10**

Generator instancji tworzy zadania iteracyjnie w pętli aż do osiągnięcia liczby **n** zadań. W celach pomocniczych została utworzona sztuczna zmienna **time**, modelująca oczekiwana czas zakończenia każdego zadania. Jest ona inkrementowana w każdej iteracji.

Dla każdego zadania zostaje wygenerowana pseudolosowo wartość wagi $\mathbf{w_j}$ z zakresu **<1, max_weight>.** Kolejnym etapem jest wytworzenie trzech operacji zadania. Rozpoczyna się to od wylosowania pseudolosowej wartości oznaczaną zmienną **length,** będącą sumą czasów trwania operacji zadania czyli $\mathbf{p_{j1} + p_{j2} + p_{j3}}$. Możliwe wartości zmiennej **length** losowane są z zakresu **<20, max_length>**.

W procesie generowanie instancji przyjęto, że jeden czas trwania operacji będzie długi tzn. z zakresu **<45% * length, 65% * length>**, jeden średni z zakresu **<25% * length, 30% * length**> oraz jeden krótki obliczany jako dopełnienie zmiennej **length**, po wylosowaniu czasu operacji długiej i średniej. Wynika z tego, iż zadanie krótkie wyraża się wzorem: $\mathbf{length - p_{długie} - p_{średnie}}$. Dłuższa operacja wybiera pseudolosowo z prawdopodobieństwem 50% maszyne $\mathbf{M_2}$ i $\mathbf{M_3}$. Po wybraniu, na której maszynie ma znajdować się zadanie długie, zadanie średnie jest przydzielane do pozostałej niewylosowanej maszyny (odpowiednio $\mathbf{M_3}$ lub $\mathbf{M_2}$). Krótkie zadanie zawsze znajduję się na maszynie $\mathbf{M_1}$.

W następnym kroku ustalana zostaje wartość oczekiwanego zakończenia zadania $\mathbf{d_j}$. Odbywa się to przy wygenerowaniu pseudolosowej wartości oraz przy użyciu jednego z dwóch wzorców:

1. Jeśli **nr iteracji** w funkcji modulo 5 wynosi 2 lub 3 (czyli co trzecie lub czwarte zadanie z kolejno generowanych pięciu zadań) to wygenerowana wartość jest równa **10% * length**. Z prawdopodobieństwem 20% wartość ta może być przemnożona przez -1.

2. W pozostałych przypadkach wybierana jest pseudolosowa wartość z zakresu **<60% * length, 110% * length>.**

Następnie wylosowana wartość z jednego ze wzorców jest dodawana do zmiennej **time**. Suma ta jest przypisywana jako wartość $\mathbf{d_j}$ generowanego zadania. Zaktualizowana wartość zmiennej **time** zostaje użyta w następnej iteracji.

Na koniec po wygenerowaniu wszystkich **n** zadań, indeks każdego zadania jest przydzielany pseudolosowo, więc prawdopodobieństwo sytuacji w której zadania znajdujące się w pliku instancji byłyby w kolejności generowania jest bliska zeru.

Niestety jak się później okazało, pomysł generacji instancji okazał się nie najlepszy, co zostało opisane dokładniej we wnioskach sprawozdania.

3. Opis algorytmu

Algorytm bazuje na rozwiązaniu listowym. Na początku sortowana jest lista wszystkich zadań rosnąco według oczekiwanego terminu zakończenia $\mathbf{d_j}$ każdego zadania.

W przypadku sortowania został użytu algorytm Quicksort posiadający średnią złożonością **O(n log n)**, a najgorszą **O($\mathbf{n^2}$)**. Po wstępnym ułożeniu zadań następuje właściwe uszeregowanie:

Założenia:

- Lista dostępnych zadań - lista posortowanych uprzednio zadań.
- Liczba **X** - liczba zadań branych pod uwagę przy wybieraniu najlepszego zadania w iteracji. Wyznaczana jest jednorazowo na początku działania algorytmu i wynosi zawsze **10 + część całkowita** (podłoga) **z dzielenia n/50**. Przykładowo dla rozmiaru instancji 50 wynosi 11, dla instancji 500 wynosi 20 itp.

Lista kroków:

1. Dopóki istnieje choć jedno nieuszeregowane zadanie (lista dostępnych zadań jest większa od 0) to kontynuuj, w przeciwnym przypadku zakończ działanie algorytmu.
2. Spośród dostępnych zadań wybierz listę **X pierwszych** zadań. Jeśli długość listy dostępnych zadań jest mniejsza niż **X** to weź wszystkie dostępne zadania.

3. Spośród wybranej listy **X** zadań wybierz zadanie o najmniejszej wartości wyrażenia: sumy długości wszystkich operacji podzielonej przez wagę zadania czyli $\mathbf{(p_{j1} + p_{j2} + p_{j3}) / w_j}$.
4. Wykonaj wszystkie operacje wybranego zadania na maszynach $\mathbf{M_1}$, $\mathbf{M_2}$ oraz $\mathbf{M_3}$.
5. Usuń wybrane zadanie z listy dostępnych zadań i powróć do **punktu 1.**

Algorytm ten faworyzuje wykonywanie zadań o niskiej wartości wyrażenia: $\mathbf{(p_{j1} + p_{j2} + p_{j3}) / w_j}$ spośród grupy zadań o najbliższym (w danym momencie) czasie oczekiwanego zakończenia $\mathbf{d_j}$. Prowadzi to do tego, że zadania o wysokiej wadze lub stosunkowo krótkie (patrząc na wszystkie czasy operacji) zostaną wykonane jako pierwsze.

Złożoność początkowego sortowania Quicksortem listy zadań wynosi **O(n log n)**. Następnie algorytm wykonuje w pętli **n** razy:

- operacje **min** po **X** zadaniach czyli **O(10 + n/50)** co sprowadza się do **O(n)**

Podsumowując złożoność algorytmu wynosi $\mathbf{O(n \log n + n^2)}$, co sprowadza się do złożoności $\mathbf{O(n^2)}$.

4. Wyniki algorytmu i obserwacje

n	Wartość kryterium dla sztucznego pliku wynikowego **A**	Wartość kryterium uzyskana przez własny algorytm **B**	Względna różnica w [%] **C = (A-B)/A *100**	Czas działania algorytmu [μs]
50	222,23	49,1	77,91	1214980
100	348,3	45,06	87,06	1051176
150	409,06	63,09	84,58	997668
200	601,48	43,14	92,83	1135725
250	843,02	52,95	93,72	1143470
300	1108,02	61,51	94,45	1218321
350	1201,84	60,21	94,99	1025727
400	1350,29	68,96	94,89	1093984
450	1613,4	71,84	95,55	974112
500	1708,01	76,45	95,52	1189425

Średnia wartość **C** dla moich instancji: **91,15%**
Średnia wartość **C** dla całej biblioteki: **64,28%**

Jak widać w przedstawionych danych algorytm osiąga bardzo duże względne różnice dla wygenerowanych instancji własnych. Są one średnio na poziomie **91,15%**. Niestety jednak nie jest to spowodowane jedynie samym działaniem algorytmu. Wpływ na takie wyniki ma sposób generowania instancji, który nie jest do końca adekwatny w stosunku do omawianego problemu. Błędnym założeniem było oparcie oczekiwanego terminu wykonania zadania $\mathbf{d_j}$ na łącznej długość wszystkich operacji tego zadania czyli $\mathbf{p_{j1} + p_{j2} + p_{j3}}$ (nazwanej poprzednio length)**.** Spowodowało to sztuczne wydłużenie wartości $\mathbf{d_j}$**,** przez co ułożenie zadań jedynie po rosnącej wartości $\mathbf{d_j}$ skutkowało, że zadania bardzo rzadko się opóźniały. W takim wypadku uszeregowanie zadań dla tych instancji okazało się bardzo łatwe. Niestety błąd został wykryty w późniejszym terminie i zgodnie z obowiązującymi zasadami nie było już możliwości poprawy generatora.

Algorytm poradził sobie w sposób zadowalający na instancjach całej biblioteki, o czym świadczy średnia różnica **C** na poziomie **64,28%**.